# **Blockchain Voting System - Tech Stack Summary**

## **Frontend**

* React: The main frontend framework used for building the user interface
* Ethers.js: Ethereum library for interacting with the blockchain, smart contracts, and wallet connections

## **Backend**

* Node.js: JavaScript runtime environment for the server
* Express.js: Web application framework for building the REST API
* Postgresql

## **Blockchain**

* Solidity: Programming language for writing smart contracts
* Ganache: Local Ethereum blockchain for development and testing
* MetaMask: Web3 wallet for user authentication and transaction signing
* Web3.js/Ethers.js: Libraries for interacting with the Ethereum blockchain

## **Authentication & Security**

* JWT (JSON Web Tokens): For user authentication and session management
* bcrypt: For password hashing
* MetaMask: For wallet-based authentication and transaction signing

# **Blockchain Voting System - Flow Summary**

## **1. User Authentication Flow**

1. Initial Connection

* User connects their MetaMask wallet
* System verifies wallet connection through WalletContext
* User's wallet address is stored and used for identification

1. Registration/Login

* New users register with email, password, and wallet address
* Existing users login with credentials
* JWT tokens are issued for session management

## **2. Voting Flow**

1. Ballot Access

* Users view available ballots on the Ballots page
* Each ballot shows:
* Title and description
* Start and end dates
* Current status (active/completed)
* Vote count

1. Voting Process

* User selects a ballot to vote on
* System checks if user is eligible to vote
* User casts their vote through the blockchain
* Vote is recorded both on-chain and in the database
* User receives confirmation of their vote

1. Vote Verification

* Votes are stored on the blockchain for transparency
* Each vote is linked to the voter's wallet address
* Users can verify their votes in the MyVotes page

## **3. Admin Flow**

1. Ballot Management

* Admins can create new ballots
* Set ballot parameters (title, description, dates)
* Monitor voting progress
* View results

1. User Management

* View registered users
* Monitor user activity
* Manage user permissions

## **4. Data Flow**

1. Frontend to Backend

* API calls for user authentication
* Ballot data retrieval
* Vote submission

1. Backend to Blockchain

* Smart contract interactions
* Vote recording
* Transaction verification

1. Blockchain to Frontend

* Vote confirmation
* Transaction status
* Real-time updates

# **Blockchain Voting System - Project Summary**

## **Core Purpose**

This is a decentralized voting system that leverages blockchain technology to ensure transparent, secure, and verifiable voting processes. The system allows users to cast votes on various ballots while maintaining the integrity and immutability of the voting records.

## **Key Components**

### **1. Smart Contract (VotingContract.sol)**

* Handles the core voting logic on the blockchain
* Manages ballot creation and voting
* Ensures one vote per wallet address
* Stores vote records immutably on the blockchain

### **2. Frontend Application**

* User Interface
* Modern, responsive design using Ant Design components
* Intuitive navigation between different sections
* Real-time updates on voting status and results